MQTT 프로토콜에서 성능향상을 고려한 Publish Queue 기반 메시지 전송 기법

임광규*, 박지수**1), 손진곤*
*한국방송통신대학교 대학원 정보과학과
**경기대학교 교양학부
e-mail:lahuman@knou.ac.kr

Publish Queue based Message Transmission Techniques considering Performance Improvement in MQTT Protocol

Kwang Kyu Lim*, JiSu Park**, Jin Gon Shon*
*Dept. of Computer Science, Graduate School, Korea National Open University
**Division of General Studies, Kyonggi University

요 약

MQTT 프로토콜은 저전력 대비 고성능으로 모바일 디바이스의 실시간 메시지 전송 시스템에 사용한다. 실시간 메시지 시스템을 구축하기 위해서는 신뢰적인 메시지 전송과 메시지간 순서 보장이 반드시 이루어져야 한다. 기존 연구에서는 MQTT 프로토콜의 QoS 2 레벨을 이용하여 메시지 순서를 보장하는 신뢰적인 메시지 시스템을 설계 및 구현하였으나, QoS 1 레벨보다 성능이 낮아진다. 따라서 본 논문에서는 MQTT 프로토콜의 QoS 1 레벨을 사용하고, Publish Queue를 이용하여 순서 보장과 함께 성능 향상을 고려한 메시지 전송 기법을 제안한다.

1. 서론

모바일 인터넷의 발전으로 모바일 기기는 인스턴스 메시지를 주고받는 주요 방법이 되었다. 그러나 독자적인 프로토콜로 구축된 메신저 애플리케이션은 확장성과 기능의 문제로 인해 표준 프로토콜 기반의 메시지 푸시 플랫폼으로 바뀌게 되었다. MQTT 프로토콜은 단순함과 편리함 때문에 많은 응용프로그램에서 사용되었다[1]. 통신사에서 제공하는 SMS, LMS, MMS는 비용이 발생하고 상대방의 메시지 확인 여부를 알려 주지 않는다[2].

MQTT(Message Queue Telemetry Transport) 프로토콜은 저전력의 고성능 프로토콜로서 모바일 메시지 전송 또는 사물과 사물 사이의 메시지 전송 애플리케이션에서 많이 사용되고 있지만, 네트워크 잠시 단절이 발생하였을 경우 단절 동안의 저장된 메시지가 동시에 유입되어 도착순서가 출발 순서와 다른 경우가 발생할 수 있다. 모바일 메신저를 사용하는 경우, 사용 편의성을 위하여 모바일 메신저는 메시지의 전송 여부를 확인할 수 있어야 하고, 메시지의 전송 순서를 보장해야 한다. 따라서 MQTT 프로토콜을 이용할 경우 수신 메시지의 순서에 대한 처리가 필요하다[3]. 기존 연구에서는 QoS 2 레벨을 이용하여 메시지 순서를 보장하는 신뢰적인 메시지 시스템을 설계 및 구현하였으나, QoS 1 레벨보다 성능이 낮아지는 단점이 있다[4].

본문에서는 MQTT 프로토콜의 QoS 1 레벨을 사용하고, Publish Queue(이하 Pub Queue)를 이용하여 순서 보장과 함께 성능 향상을 고려한 메시지 전송 기법을 제안한다.

2. 관련 연구

2.1 메시지 전송 방식

인터넷을 이용하여 메시지를 전송하는 방식에는 폴링(polling) 방식과 푸시(push) 방식이 있다. 폴링 방식은 모바일 디바이스에서 주기적으로 서버에 신규 메시지가 존재하는지를 확인하여 신규 메시지가 존재하면 모바일 기기로 수신한다. 이는 주기적으로 서버에 접속해야 하고, 불필요하게 네트워크 데이터를 낭비한다는 단점을 가진다. 푸시 방식은 서버에서 신규 데이터가 발생할 경우에 각 모바일 디바이스로 전송한다. 이는 폴링 방식보다 저비용으로 구현할 수 있어 많은 메시지 전송 시스템에서 사용되는 방식이다. 푸시 방식은 XMPP와 HTTP, MQTT 그리고 CoAP 등의 프로토콜을 이용한 메시지 전송 시스템에서 사용되고 있다. 이와 같은 메시지 전송 시스템은 인터넷 기반 프로토콜과 무선전화망 기반 프로토콜로 구분한다. 이 가운데 MQTT는 대표적인 푸시 방식의 무선전

1) 교신저자

* 이 논문은 2017년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. NRF-2017R1D1A1B03035833)

화망 기반 프로토콜로써 저전력이고 고성능이라는 특징을 가지고 있어 많은 모바일 전송 시스템에서 사용되고 있다[5][6].

2.2 MQTT

MQTT 프로토콜은 1999년 IBM에서 발표한 메시지 푸시 프로토콜이다. MQTT 프로토콜은 스마트폰과 같은 제한된 컴퓨팅 성능과 빈약한 네트워크 연결 환경에서 동작하도록 설계된 대용량 메시지 전달 프로토콜로 낮은 대여폭과 긴 지연 시간에도 불구하고 신뢰적인 메시지를 전송하도록 설계되었다. MQTT 프로토콜은 2013년 OASIS(Organization for the Advancement of Structured Information Standards)에서 사물 인터넷을 위한 메시지 프로토콜로 선정하였으며, 다양한 인스턴스 메신저에서 사용된다[5].

MQTT 프로토콜은 경량의 메시지 전송 프로토콜로써 하나의 중계인(이하 Broker)서버와 두 종류의 클라이언트로 구성된다. Broker 서버는 특정 토픽에 대해 메시지를 발행하는 클라이언트(Publish 이하 Pub)와 구독하는 클라이언트(Subscribe 이하 Sub) 사이에 중계인으로서 역할을 하고, 특정 주제에 대해 다자간 채팅이나 메시지를 서로 공유하는 경우에 구독자는 구독(Sub)을 하고, 발행자는 발행(Pub)하는 구조이다. MQTT에서는 QoS 레벨을 0, 1, 2의 세 단계로 정의한다[7].

QoS 0: 한 번만 전달하고 전달 여부는 확인하지 않음

QoS 1: 적어도 한 번 이상 전달하고 전달 여부 확인

QoS 2: 4단계의 핸드 셰이킹(handshaking)을 통해 정확히 한 번만 전달(신뢰적)

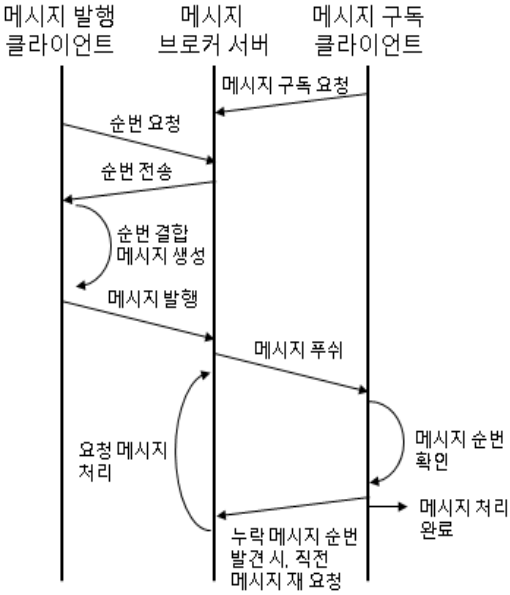
발행자와 구독자 모두 QoS를 지정할 수 있으나 발행자가 지정한 최대 QoS 레벨이 우선시 된다.

QoS 1 레벨 이상에서는 MQTT에서 제공되는 deliveryComplete[10]을 이용하여 메시지 전송 성공 여부가 확인 가능하다. 하지만 QoS 1 레벨은 메시지가 한 번 이상을 전달될 수 있기 때문에 응용프로그램에서 메시지에 대한 중복에 대한 처리를 구현해야 한다[8].

2.3 기존 연구 분석

기존의 메시지 푸시 프로토콜 가운데 MQTT 프로토콜이 모바일 디바이스 환경에서 적은 자원으로 사용 가능하며, QoS 레벨을 이용해서 단일 메시지의 전송 여부를 보장할 수 있는 것을 확인한다. 그러나 메시지 전송 프로토콜은 단일 메시지에 대한 메시지 전송 여부만 보장하며, 다수의 메시지가 전송될 때 메시지 간의 순차적인 전송을 보장하지 하는 문제를 가진다.

기존 연구에서는 QoS 2 레벨의 MQTT 프로토콜 기반으로 신뢰적인 메시지 전송 시스템을 설계하고 구현하였다. 이는 (그림 1)과 같이 신뢰적인 메시지 전송 시스템에 대한 요구 사항을 단일 메시지의 전송 여부와 메시지 사이에 순번이 보장되어 메시지가 순차적으로 도착하고, 누락 메시지를 발견한 상황에 해당 메시지를 재요청하여 수신 받을 수 있도록 하였다. 또한 신뢰적인 메시지 전송 시스템은 메시지를 전송할 때 순차적이며 긴 지연 시간 없이 메시지가 전송한다. 그리고 MQTT 프로토콜만을 이용하였을 때 메시지가 휘발성으로 삭제되는 기존의 메시지와 달리 재사용을 위한 메시지가 데이터베이스에 저장된다. 기존 연구의 최종 실험에서는 MQTT 프로토콜만을 사용했을 때보다 서버의 자원 사용이 증가하나, 폴링 방식보다는 저비용이며 메시지가 신뢰적으로 전송된다[4].



(그림 1) 신뢰적인 메시지 전송 시스템

3. Pub Queue 기반 메시지 전송 기법

3.1 MQTT 프로토콜 데이터 구조

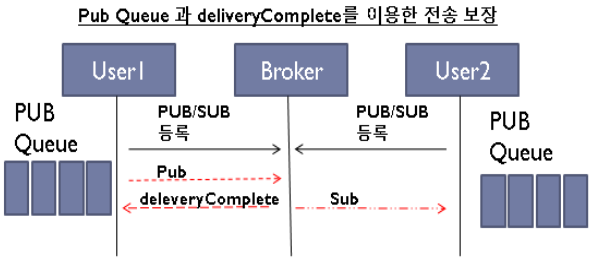
<표 1> MQTT 프로토콜 데이터 구조

Key	Value
topic1/group1/user1	[{“id”:currentmillis,“msg”:“hello word”, ″sender″:″user2″}, {“id”:currentmillis,“msg”:“hello word”, ″sender″:″user1″}]
topic1/group2/#	
topic2/group1/user2	
topic2/group2/#	
topic1/group2/#	

데이터 구조는 Key/Value 형식이다. 기본적으로 특정키에 데이터를 저장하는 방식으로 되어 있다. Key 데이터는 <표 1>과 같은 구조를 가지게 된다.

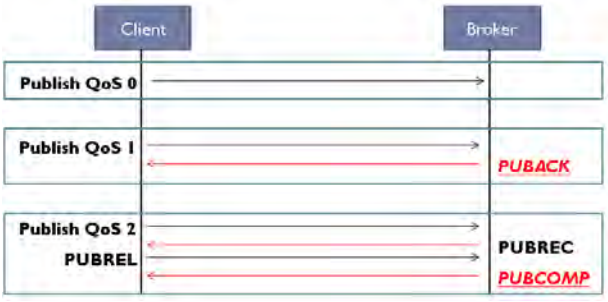
Key는 MQTT에서 주제로 묶여 발행되는 토픽으로 / (슬러쉬)로 이루어진 계층 구조를 가진다. 메시지의 경우 JSON 형태를 띄우며, id 값으로 currentmillis[9] 값을 사용하여 Pub에서 Broker로, Broker에서 Sub으로 전송한 시간을 확인한다.

3.2 Pub Queue 생성과 메시지 흐름



(그림 2) Pub Queue 생성과 Broker 등록

User1과 User2에서 모바일 애플리케이션이 구동되면 메시지 전송을 위한 한 개의 Pub Queue를 생성하고, 사용자가 메시지 전송 시 Pub Queue에 저장을 한다. 메시지의 전송 성공이 deliveryComplete를 통해 확인되면 다음 메시지를 전송한다. deliveryComplete 기능은 QoS 1 레벨이상일 경우만 사용이 가능하다. MQTT 프로토콜의 스펙에 따라 (그림 3)와 같이 QoS 1 레벨의 경우 PUBACK이 수신되면 deliveryComplete가 호출되고, QoS 2 레벨의 경우 PUBCOMP 가 수신되면 deliveryComplete가 호출된다. delieveryComplete를 이용해 Broker로 메시지 전송의 성공 시 다음 메시지를 Pub Queue에서 추출하여 Broker로 전송한다.

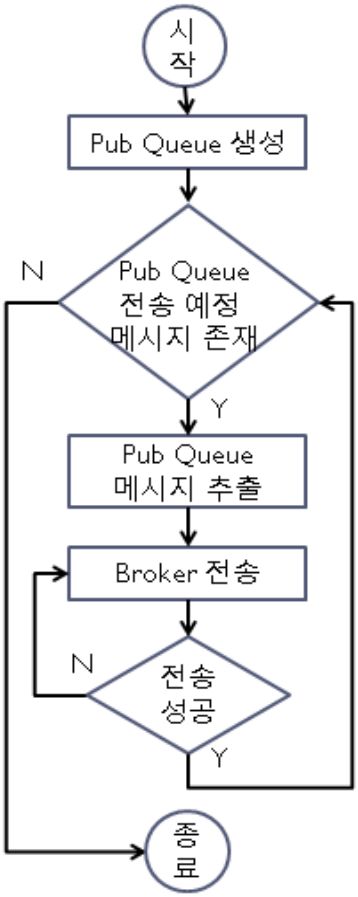


(그림 3) deliveryComplete 호출 시점

Sub 클라이언트에서는 수신되는 메시지의 순서가 섞여서 데이터를 수신 할 경우가 발생한다. 이때, 클라이언트에서 id 값인 currentmillis를 기준으로 수신된 메시지의 id가 기존 id 보다 작을 경우 해당 메시지의 위치를 기존 id 보다 상위에 표출하도록 순서 보정을 하여 화면에 출력한다.

3.3 메시지 전송 순서도

Pub Queue 기반의 메시지 전송은 Pub Queue를 생성하고 메시지 유입이 되면 한 개의 메시지를 추출하여 Broker로 전송한다. 이때 전송이 실패하게 되면 Broker에 추출한 메시지를 재전송한다. 전송에 성공할 경우 Pub Queue에 다음 메시지 존재 여부를 확인하고 존재할 경우 다음 메시지를 추출하고 전송하는 과정을 되풀이한다. Broker에서는 id 값으로 중복 확인이 가능 하다. 다음 (그림 4)는 메시지 전송 순서를 나타낸다.



(그림 4) 메시지 전송 순서도

4. 시뮬레이션

4.1 실험 환경

실험 환경은 <표 2>와 같은 소프트웨어 환경과 하드웨어 환경으로 구성한다. 하드웨어로는 Broker와 한 대와 클라이언트(Pub/Sub) 두 대로 구성하였다. 모든 메시지 크기는 동일하게 26 bytes이며 0001~1000 까지 메시지 내에 순서를 가지고 있다.

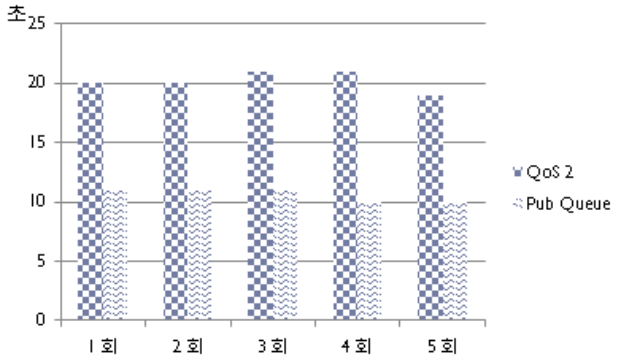
<표 2> 구현 환경

소프트웨어 환경	
OS	Windows 10
Message Broker	mosquitto
MQTT Library	Eclipse Paho
Dev. Lenguage	JAVA
하드웨어 환경	
CPU	Intel Core i3-4300
Memory	8 Gbyte

4.2 실험 결과

실험은 QoS 1레벨에서 Pub Queue 기반 메시지 전송 기법과 QoS 2 레벨과 비교하여 성능을 비교한다.

QoS 1 레벨로 Pub Queue를 생성하고 Broker에 등록 후 메시지를 전송하는 테스트와 QoS 2 레벨로 Broker에 메시지를 전송하는 테스트를 1,000건의 메시지로 각 5회씩 진행하였다.



(그림 5) QoS 2 와 Pub Queue의 성능 평가

실험 결과에서는 (그림 5)과 같이 QoS 1 레벨의 Pub Queue를 사용한 결과가 QoS 2 레벨을 사용한 테스트 결과보다 약 2배의 성능 차이를 보였다. 또한 Pub Queue와 deliveryComplate를 이용하여 메시지를 전송 시 0001~1000 까지 메시지 누락 없이 순서대로 Broker 도착을 확인하였다.

5. 결론

기존 연구에서는 MQTT 프로토콜의 QoS 2 레벨을 이용하여 메시지 순서를 보장하는 신뢰적인 메시지 시스템을 설계 및 구현하였으나, QoS 1 레벨보다 성능이 낮아지는 문제점이 있다. 이를 해결하기 위해 본 논문에서는 MQTT 프로토콜의 QoS 1 레벨을 사용하고, Pub Queue를 이용하여 전송 보장과 함께 성능 향상을 고려한 메시지 전송 기법을 제안하였다. 성능 평가를 통하여 QoS 1 레벨을 사용하였을 경우 QoS 2 레벨보다 성능이 최대 2배 이상 향상된 것을 성능 평가를 통해 확인하였다.

서비스 운영에서는 네트워크 트래픽과 제한된 자원을 이유로 QoS 2 레벨보다 QoS 1 레벨에서 안정적으로 메시지를 전송하는 것이 중요하다. 실제 예로 대표적인 클라우드 서비스인 AWS, AZURE, GOOGLE CLOUD에서는 MQTT 프로토콜의 QoS 2 레벨을 지원하지 않는다.

참고문헌

[1] Ma Yue, Yan Ruiyang, Sun Jianwei, Yao Kaifeng, "A MQTT Protocol Message Push Server Based on RocketMQ," 2017 10th International Conference on Intelligent Computation Technology and Automation (ICICTA), pp.295-298, 2017.

[2] "SKT 문자요금", Tworld.co.kr, 2017. [Online]. Available: http://www.tworld.co.kr/normal.do?serviceId=S_PROD2001&viewId=V_PROD2001&prod_id=NA00003036. [Accessed: 05- Sep- 2017].

[3] "Facebook 메신저와 MQTT", D2.naver.com, 2017. [Online]. Available: http://d2.naver.com/helloworld/1846. [Accessed: 05- Sep- 2017].

[4] 황현천. "MQTT 프로토콜 기반의 신뢰적인 메시지 전송 시스템의 설계 및 구현", 한국방송통신대학교 대학원 석사학위 논문, 2016

[5] 심승현, 김학범, "사물인터넷과 MQTT 기술", 정보보호학회지, 제 24권 제 6호, 2014, 12

[6] Shinho Lee, Hyeonwoo Kim, Dong-kweon Hong, Hongtaek Ju,"Correlation analysis of MQTT loss and delay according to QoS level," The International Conference on Information Networking 2013 (ICOIN), pp.714 - 717, 2013.

[7] P. Th. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec. "The many faces of publish/subscribe," J. ACM Comput. Surveys(CSUR), vol. 35, no. 2, pp. 114-131, June 2003.

[8] "MQTT 프로토콜 분석과 테스트 | Hard Copy Arduino", Hardcopyworld.com, 2017. [Online]. Available: http://www.hardcopyworld.com/ngine/aduino/index.php/archives/2562. [Accessed: 05- Sep- 2017].

[9] T. Sandman, "Current Millis", Current Millis, 2017. [Online]. Available: https://currentmillis.com/. [Accessed: 05- Sep- 2017].

[10] "MQTTv3", javadoc, 2017. [Online]. Available: https://www.eclipse.org/paho/files/javadoc/org/eclipse/paho/client/mqttv3/MqttCallback.html#deliveryComplete-org.eclipse.paho.client.mqttv3.IMqttDeliveryToken-. [Accessed: 07- Nov- 2017].